Subject: Pixmap Problems
Posted by Kenneth Kump on Thu, 05 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Howdy,

I'm working on a SUNOS IDL version 4.0.1 on an 8bit display with a shared color table giving about 243 colors. I have slightly modified the Black/white linear colortable and have set the color of !d.n_colors-1=RED using tvlct.

Problem: Using Pixmaps I have discovered 2 problems

1) I have created a "mask" pixmap. It consists of 0's and 255's. The intention is to display an image, then do device,set_graphics=1 (this is an AND) then do device,copy=[...]. What happens is strange. The values which are to be masked to Zero get a displayed color which is very odd and random (looks like from private color table). Non masked values are fine. When I do a tvrd() it has the correct values, and I can re-tv this captured image and it is perfect.

2) I create pixmap and tv a value of !d.n_colors-1 (RED). Now I do a device,copy=[] (regular graphics mode) and what I get is white instead of red.

What's going on? I need the speed of a pixmap, but results are unsatisfactory.

Any hints appreciated,

-Ken Kump

Subject: Re: Pixmap Problems
Posted by steinhh on Mon, 30 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Path: medusa.uio.no!steinhh
Newsgroups: comp.lang.idl-pvwave
Sender: steinhh@amon (Stein Vidar Hagfors Haugan)
From: steinhh@amon.uio.no (Stein Vidar Hagfors Haugan)
Distribution: world
References: <322EDB2A.167EB0E7@scr.siemens.com>
Reply-To: steinhh@amon.uio.no (Stein Vidar Hagfors Haugan)

Organization: Institute for Theoretical Astrophysics
Subject: Re: Pixmap Problems

In article <322EDB2A.167EB0E7@scr.siemens.com>, Kenneth Kump
<kkump@scr.siemens.com> writes:

|> 1) I have created a "mask" pixmap. It consists of 0's and
|> 255's. The intention is to display an image, then do
|> device,set_graphics=1 (this is an AND) then do
|> device,copy=[...]. What happens is strange. The values which
|> are to be masked to Zero get a displayed color which is
|> very odd and random (looks like from private color table).
|> Non masked values are fine. When I do a tvrd() it has the
|> correct values, and I can re-tv this captured image and
|> it is perfect.
|>
|>
|> 2) I create pixmap and tv a value of !d.n_colors-1 (RED).
|> Now I do a device,copy=[] (regular graphics mode)
|> and what I get is white instead of red.
|>
|>
|> What's going on? I need the speed of a pixmap, but results are
|> unsatisfactory.

Whenever IDL doesn't grab it's own private colormap, the byte values
actually stored in video memory are not necessarily in the range
(0, !D.N_colors-1), since IDL may not have control over those
"physical" colors. Instead, IDL tries to hide this from the user,
translating between "IDL colors" and "video colors" (also when reading
back with tvrd()).

This sometimes gives odd results when using the graphics functions
the way they apparently should work. E.g., color zero could correspond
to a non-zero value, so doing an XOR with color 0 does not necessarily
leave values untouched (the graphics functions seem to be applied *after*
the translation to physical color numbers).

I seem to remember reading some stuff about this in the reference guide.

From your example #2, it seems IDL doesn't cope well with that translation
with regard to pixmaps. If you can fool IDL into translating the values
correctly (or doing it yourself) when they are put into the pixmap,
you should be OK as to this problem.

Stein Vidar

Subject: Re: pixmap problem
Posted by R.Bauer on Thu, 28 Aug 2003 07:15:16 GMT
View Forum Message <> Reply to Message

Steve Ready wrote:

Dear Steve,

in addition to total what shows min and max.

Reimar

> Folks,
>
> I am hoping someone can shed some light on this problem.
>
> I am creating a large image in graphics memory with WINDOW,/PIXMAP and
> drawing to it using the PLOTS routine. I have discovered that if I specify
> a pixmap size larger than a particular value, dependent on the graphics
> card, I am able to allocate the graphics memory with no problem but am not
> able to draw to it. I have verified this on an WinXP and Win2K machine,
> both with 32mb graphics cards. Sample demo test code follows with typical
> output. This is slightly modified code from RSI website for testing
> available graphics memory size. Any clues?
>
> Thanks, Steve
>
> Steve Ready
> Sr. Member of the Research Staff
> Electronic Materials Lab
> Palo Alto Research Center
> 3333 Coyote Hill Rd.
> Palo Alto, CA 94034
> Voice: 650-812-4135
> FAX: 650-812-4105
> Email: ready@parc.com
> http://www.parc.com/ready
>
> ;** ***
> PRO test_pixmap_size
>
> cnt = 40L
> increment = 100
> off=3000
> i = 1
>
> ; Catch when the creation of a pixmap
> ; fails, and report the previous
> ; pixmap dimensions that succeeded.

```
>
>  CATCH, errStat
>  IF (errStat NE 0) THEN BEGIN
>     x = ((i-1)*increment)+off
>     y = x
>     PRINT, 'Suggested maximum pixmap size: ', x, ' by ', y
>     RETURN
>  ENDIF
>
>  ; Loop through potential pixmap dimensions.
>  FOR i=1,cnt DO BEGIN
>
>     x = (i*increment)+off
>     y = (i*increment)+off
>     print, 'Trying: ', x, ' by ', y
>
>     WINDOW, /PIXMAP, /FREE, XSIZE=x, YSIZE=y
>     plots,[.5,.5],[.5,.5],/normal
>     print, total(tvrd())
>     WDELETE, !D.WINDOW
>
>  ENDFOR
>  END
>   ;************************************************************ ***
>
>  Result is:
>
>  IDL> test_pixmap_size
>  Trying: 3100 by 3100
>  255.000
>  Trying: 3200 by 3200
>  255.000
>  Trying: 3300 by 3300
>  255.000
>  Trying: 3400 by 3400
>  255.000
>  Trying: 3500 by 3500
>  0.000000
>  Trying: 3600 by 3600
>  0.000000
>  Trying: 3700 by 3700
>  0.000000
>  Trying: 3800 by 3800
>  0.000000
>  Trying: 3900 by 3900
>  0.000000
>  Trying: 4000 by 4000
>  0.000000
```

> Trying: 4100 by 4100
> 0.000000
> Trying: 4200 by 4200
> 0.000000
> Trying: 4300 by 4300
> 0.000000
> Trying: 4400 by 4400
> 0.000000
> Trying: 4500 by 4500
> 0.000000
> Trying: 4600 by 4600
> 0.000000
> Trying: 4700 by 4700
> 0.000000
> Trying: 4800 by 4800
> 0.000000
> Trying: 4900 by 4900
> Suggested maximum pixmap size: 4800 by 4800

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
 ========== ======
a IDL library at ForschungsZentrum Juelich
 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

---

Subject: Re: pixmap problem
Posted by Steve Ready on Thu, 28 Aug 2003 16:36:44 GMT
View Forum Message <> Reply to Message

Well, min would be 0 and max would be 255 for the pixmaps that worked and 0
for the ones that didn't. Since the PLOTS command only turns on one pixel in
the image, total=max for them all.

Steve

"Reimar Bauer" <R.Bauer@fz-juelich.de> wrote in message
news:bika7t$1g9f$1@zam602.zam.kfa-juelich.de...
> Steve Ready wrote:
>
> Dear Steve,
>
> in addition to total what shows min and max.
>
> Reimar

>
>> Folks,
>>
>> I am hoping someone can shed some light on this problem.
>>
>> I am creating a large image in graphics memory with WINDOW,/PIXMAP and
>> drawing to it using the PLOTS routine. I have discovered that if I
specify
>> a pixmap size larger than a particular value, dependent on the graphics
>> card, I am able to allocate the graphics memory with no problem but am
not
>> able to draw to it. I have verified this on an WinXP and Win2K machine,
>> both with 32mb graphics cards. Sample demo test code follows with
typical
>> output. This is slightly modified code from RSI website for testing
>> available graphics memory size. Any clues?
>>
>> Thanks, Steve
>>
>> Steve Ready
>> Sr. Member of the Research Staff
>> Electronic Materials Lab
>> Palo Alto Research Center
>> 3333 Coyote Hill Rd.
>> Palo Alto, CA 94034
>> Voice: 650-812-4135
>> FAX: 650-812-4105
>> Email: ready@parc.com
>> http://www.parc.com/ready
>>
>> ;** ***
>> PRO test_pixmap_size
>>
>> cnt = 40L
>> increment = 100
>> off=3000
>> i = 1
>>
>> ; Catch when the creation of a pixmap
>> ; fails, and report the previous
>> ; pixmap dimensions that succeeded.
>>
>> CATCH, errStat
>> IF (errStat NE 0) THEN BEGIN
>> x = ((i-1)*increment)+off
>> y = x
>> PRINT, 'Suggested maximum pixmap size: ', x, ' by ', y
>> RETURN

```
>>  ENDIF
>>
>>  ; Loop through potential pixmap dimensions.
>>  FOR i=1,cnt DO BEGIN
>>
>>    x = (i*increment)+off
>>    y = (i*increment)+off
>>    print, 'Trying: ', x, ' by ', y
>>
>>    WINDOW, /PIXMAP, /FREE, XSIZE=x, YSIZE=y
>>    plots,[.5,.5],[.5,.5],/normal
>>    print, total(tvrd())
>>    WDELETE, !D.WINDOW
>>
>>  ENDFOR
>>  END
>>   ;************************************************************ ***
>>
>>  Result is:
>>
>>  IDL> test_pixmap_size
>>  Trying: 3100 by 3100
>>  255.000
>>  Trying: 3200 by 3200
>>  255.000
>>  Trying: 3300 by 3300
>>  255.000
>>  Trying: 3400 by 3400
>>  255.000
>>  Trying: 3500 by 3500
>>  0.000000
>>  Trying: 3600 by 3600
>>  0.000000
>>  Trying: 3700 by 3700
>>  0.000000
>>  Trying: 3800 by 3800
>>  0.000000
>>  Trying: 3900 by 3900
>>  0.000000
>>  Trying: 4000 by 4000
>>  0.000000
>>  Trying: 4100 by 4100
>>  0.000000
>>  Trying: 4200 by 4200
>>  0.000000
>>  Trying: 4300 by 4300
>>  0.000000
>>  Trying: 4400 by 4400
```

>> 0.000000
>> Trying: 4500 by 4500
>> 0.000000
>> Trying: 4600 by 4600
>> 0.000000
>> Trying: 4700 by 4700
>> 0.000000
>> Trying: 4800 by 4800
>> 0.000000
>> Trying: 4900 by 4900
>> Suggested maximum pixmap size: 4800 by 4800
>
> --
> Forschungszentrum Juelich
> email: R.Bauer@fz-juelich.de
> http://www.fz-juelich.de/icg/icg-i/
> ========== ======
> a IDL library at ForschungsZentrum Juelich
> http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
>

Subject: Re: pixmap problem
Posted by R.Bauer on Thu, 28 Aug 2003 16:53:15 GMT
View Forum Message <> Reply to Message

Steve Ready wrote:

> Well, min would be 0 and max would be 255 for the pixmaps that worked and
> 0 for the ones that didn't. Since the PLOTS command only turns on one
> pixel in the image, total=max for them all.
>
> Steve

Dear Steve,

I asked because my total in 6.0beta does not work right.
I got extremly large numbers while min max gives the right value.

I don't know what's the official version did or if I have found a new bug.

Reimar

>
>
> "Reimar Bauer" <R.Bauer@fz-juelich.de> wrote in message

> news:bika7t$1g9f$1@zam602.zam.kfa-juelich.de...
>> Steve Ready wrote:
>>
>> Dear Steve,
>>
>> in addition to total what shows min and max.
>>
>> Reimar
>>
>>> Folks,
>>>
>>> I am hoping someone can shed some light on this problem.
>>>
>>> I am creating a large image in graphics memory with WINDOW,/PIXMAP and
>>> drawing to it using the PLOTS routine. I have discovered that if I
> specify
>>> a pixmap size larger than a particular value, dependent on the graphics
>>> card, I am able to allocate the graphics memory with no problem but am
> not
>>> able to draw to it. I have verified this on an WinXP and Win2K machine,
>>> both with 32mb graphics cards. Sample demo test code follows with
> typical
>>> output. This is slightly modified code from RSI website for testing
>>> available graphics memory size. Any clues?
>>>
>>> Thanks, Steve
>>>
>>> Steve Ready
>>> Sr. Member of the Research Staff
>>> Electronic Materials Lab
>>> Palo Alto Research Center
>>> 3333 Coyote Hill Rd.
>>> Palo Alto, CA 94034
>>> Voice: 650-812-4135
>>> FAX: 650-812-4105
>>> Email: ready@parc.com
>>> http://www.parc.com/ready
>>>
>>> ;** ***
>>> PRO test_pixmap_size
>>>
>>> cnt = 40L
>>> increment = 100
>>> off=3000
>>> i = 1
>>>
>>> ; Catch when the creation of a pixmap
>>> ; fails, and report the previous

```
>>>  ; pixmap dimensions that succeeded.
>>>
>>>  CATCH, errStat
>>>  IF (errStat NE 0) THEN BEGIN
>>>     x = ((i-1)*increment)+off
>>>     y = x
>>>     PRINT, 'Suggested maximum pixmap size: ', x, ' by ', y
>>>     RETURN
>>>  ENDIF
>>>
>>>  ; Loop through potential pixmap dimensions.
>>>  FOR i=1,cnt DO BEGIN
>>>
>>>     x = (i*increment)+off
>>>     y = (i*increment)+off
>>>     print, 'Trying: ', x, ' by ', y
>>>
>>>     WINDOW, /PIXMAP, /FREE, XSIZE=x, YSIZE=y
>>>     plots,[.5,.5],[.5,.5],/normal
>>>     print, total(tvrd())
>>>     WDELETE, !D.WINDOW
>>>
>>>  ENDFOR
>>>  END
>>>   ;************************************************************ ***
>>>
>>>  Result is:
>>>
>>>  IDL> test_pixmap_size
>>>  Trying: 3100 by 3100
>>>  255.000
>>>  Trying: 3200 by 3200
>>>  255.000
>>>  Trying: 3300 by 3300
>>>  255.000
>>>  Trying: 3400 by 3400
>>>  255.000
>>>  Trying: 3500 by 3500
>>>  0.000000
>>>  Trying: 3600 by 3600
>>>  0.000000
>>>  Trying: 3700 by 3700
>>>  0.000000
>>>  Trying: 3800 by 3800
>>>  0.000000
>>>  Trying: 3900 by 3900
>>>  0.000000
>>>  Trying: 4000 by 4000
```

>>> 0.000000
>>> Trying: 4100 by 4100
>>> 0.000000
>>> Trying: 4200 by 4200
>>> 0.000000
>>> Trying: 4300 by 4300
>>> 0.000000
>>> Trying: 4400 by 4400
>>> 0.000000
>>> Trying: 4500 by 4500
>>> 0.000000
>>> Trying: 4600 by 4600
>>> 0.000000
>>> Trying: 4700 by 4700
>>> 0.000000
>>> Trying: 4800 by 4800
>>> 0.000000
>>> Trying: 4900 by 4900
>>> Suggested maximum pixmap size: 4800 by 4800
>>
>> --
>> Forschungszentrum Juelich
>> email: R.Bauer@fz-juelich.de
>> http://www.fz-juelich.de/icg/icg-i/
>> ========== ======
>> a IDL library at ForschungsZentrum Juelich
>> http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
>>

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

---

Subject: Re: pixmap problem
Posted by Karl Schultz on Thu, 28 Aug 2003 17:13:46 GMT
View Forum Message <> Reply to Message

"Steve Ready" <ready@parc.com> wrote in message
news:bij1ge$t6r$1@news.parc.xerox.com...
> Folks,
>
> I am hoping someone can shed some light on this problem.
>

> I am creating a large image in graphics memory with WINDOW,/PIXMAP and
> drawing to it using the PLOTS routine. I have discovered that if I specify
a
> pixmap size larger than a particular value, dependent on the graphics
card,
> I am able to allocate the graphics memory with no problem but am not able
to
> draw to it. I have verified this on an WinXP and Win2K machine, both with
> 32mb graphics cards. Sample demo test code follows with typical output.
This
> is slightly modified code from RSI website for testing available graphics
> memory size. Any clues?
>
> Thanks, Steve

This is a known problem, # 22066 should you call IDL tech support about it.

There is no clear resolution to the problem, but it does appear to be pretty sensitive to the graphics card and/or driver software. On many machines, the attempt to create a "too large" pixmap will fail and any pixmap that is created successfully returns the correct data on a TVRD; this is the correct behavior. However, several users have reported the same symptoms as you are describing. If you can report what graphics hardware and driver software (including version information) you are using, perhaps we can identifiy a pattern. (And I suppose the canned response of "check your drivers for an update" is a good thing to offer here as well.)

Another strange behavior on the failing hardware is that the ORDER keyword can affect the pixmap size threshhold where the data is not read back correctly via TVRD. IDL itself does not perform the "flip" as controlled by ORDER. IDL instead modifies parameters to a WIN32 GDI call, thus making the GDI or graphics driver perform the flip. A change in behavior like this is somewhat suggestive of a driver problem.

Karl

Subject: Re: pixmap problem
Posted by Steve Ready on Thu, 28 Aug 2003 18:47:04 GMT
View Forum Message <> Reply to Message

I have replicated this with a Radion VE 32mb, Invidia GForce 64mb, and a Matrox 32mb under Win2k and XP. The maximum image size returned by the correct behavior for all these boards seems to scale loosely with the color resolution (16 versus 32 bit) coupled with the amount of card memory. It would be nice to have a better understanding of this and have IDL correctly report the limitiations.

I have reported this to RSI and here was their response:

****
We have no reason to believe that you are running into any issue other than
the limits imposed by the Windows O.S. in its interface with your graphics
card. One way to see this would be to try to make a bigger PIXMAP with the
Windows Visual C++/MFC API. In both cases, you would want to make sure that
both IDL and the MFC comparative program would have a "virgin" O.S.
environment; that is, there would not be other applications running about in
the background making demands on the video card memory. Consider that video
card memory could get fragmented just like process memory.

IDL's WINDOWS, /PIXMAP call is making a basic call on the Win32 API pixmap
library. I do not know the underlying source code for Microsoft's library,
but it is very possible that that memory automatically makes a 3x or 4x
pixmap-coordinates demand on the video card. Perhaps it does this only when
your display is set to true color; you could test and see if setting your
display to 256-colors provided you the bigger pixmap you want.

We would expect the larger memory video card to allow for a much larger
maximum PIXMAP size, but, to be certain, you may need to consult with the
video card vendor about what THEY think the maximum pixmap is that Windows
can make on their card.

****

"Karl Schultz" <kschultz_no_spam@rsinc.com> wrote in message
news:vksds6nnjrv1a@corp.supernews.com...
>
> This is a known problem, # 22066 should you call IDL tech support about
it.
>
> There is no clear resolution to the problem, but it does appear to be
pretty
> sensitive to the graphics card and/or driver software. On many machines,
> the attempt to create a "too large" pixmap will fail and any pixmap that
is
> created successfully returns the correct data on a TVRD; this is the
correct
> behavior. However, several users have reported the same symptoms as you
are
> describing. If you can report what graphics hardware and driver software
> (including version information) you are using, perhaps we can identifiy a
> pattern. (And I suppose the canned response of "check your drivers for an
> update" is a good thing to offer here as well.)
>

> Another strange behavior on the failing hardware is that the ORDER keyword
> can affect the pixmap size threshhold where the data is not read back
> correctly via TVRD. IDL itself does not perform the "flip" as controlled
by
> ORDER. IDL instead modifies parameters to a WIN32 GDI call, thus making
the
> GDI or graphics driver perform the flip. A change in behavior like this
is
> somewhat suggestive of a driver problem.
>
> Karl
>
>

Subject: Re: pixmap problem
Posted by Karl Schultz on Thu, 28 Aug 2003 20:35:16 GMT
View Forum Message <> Reply to Message

"Steve Ready" <ready@parc.com> wrote in message
news:bilina$4ee$1@news.parc.xerox.com...
> I have replicated this with a Radion VE 32mb, Invidia GForce 64mb, and a
> Matrox 32mb under Win2k and XP. The maximum image size returned by the
> correct behavior for all these boards seems to scale loosely with the
color
> resolution (16 versus 32 bit) coupled with the amount of card memory. It
> would be nice to have a better understanding of this and have IDL
correctly
> report the limitiations.
>
> I have reported this to RSI and here was their response:
>
> **
> We have no reason to believe that you are running into any issue other
than
> the limits imposed by the Windows O.S. in its interface with your graphics
> card. One way to see this would be to try to make a bigger PIXMAP with the
> Windows Visual C++/MFC API. In both cases, you would want to make sure
that
> both IDL and the MFC comparative program would have a "virgin" O.S.
> environment; that is, there would not be other applications running about
in
> the background making demands on the video card memory. Consider that
video
> card memory could get fragmented just like process memory.
>
> IDL's WINDOWS, /PIXMAP call is making a basic call on the Win32 API pixmap
> library. I do not know the underlying source code for Microsoft's library,

> but it is very possible that that memory automatically makes a 3x or 4x
> pixmap-coordinates demand on the video card. Perhaps it does this only
when
> your display is set to true color; you could test and see if setting your
> display to 256-colors provided you the bigger pixmap you want.
>
> We would expect the larger memory video card to allow for a much larger
> maximum PIXMAP size, but, to be certain, you may need to consult with the
> video card vendor about what THEY think the maximum pixmap is that Windows
> can make on their card.
>
> **********
>
>
>
> "Karl Schultz" <kschultz_no_spam@rsinc.com> wrote in message
> news:vksds6nnjrv1a@corp.supernews.com...
>>
>> This is a known problem, # 22066 should you call IDL tech support about
> it.
>>
>> There is no clear resolution to the problem, but it does appear to be
> pretty
>> sensitive to the graphics card and/or driver software. On many
machines,
>> the attempt to create a "too large" pixmap will fail and any pixmap that
> is
>> created successfully returns the correct data on a TVRD; this is the
> correct
>> behavior. However, several users have reported the same symptoms as you
> are
>> describing. If you can report what graphics hardware and driver
software
>> (including version information) you are using, perhaps we can identifiy
a
>> pattern. (And I suppose the canned response of "check your drivers for
an
>> update" is a good thing to offer here as well.)
>>
>> Another strange behavior on the failing hardware is that the ORDER
keyword
>> can affect the pixmap size threshhold where the data is not read back
>> correctly via TVRD. IDL itself does not perform the "flip" as
controlled
> by
>> ORDER. IDL instead modifies parameters to a WIN32 GDI call, thus making
> the
>> GDI or graphics driver perform the flip. A change in behavior like this

> is
>> somewhat suggestive of a driver problem.
>>
>> Karl
>>
>>
>
>

OK, I was finally able to reproduce it here - I was not able to before with the machines I had available to me.

Windows is reporting an out-of-memory condition when IDL asks it to copy the image from the pixmap (video memory) to an IDL array (system memory). One would think that this copy would usually succeed since the memory has already been allocated in both places. The pixmap has been successfully created and drawn to in video memory and the target array in system memory already exists.

The problem is that there is an intermediate copy step. I think that Windows must allocate some special GDI system pool memory that is locked or 'pinned' so it can't be swapped, presumably for a DMA, AGP, or other special memory transfer from the video memory to the pinned system memory. If GDI resources are scarce or fragmented, this allocation may fail, causing the copy to fail. I experimented quite a bit and noted some unusual patterns of success and failure, which I attribute to varying degrees of fragmentation.

I don't think that IDL can predict a failure here, but we can throw an error on failure rather than returning an array of zeros. Or, we can do the copy in smaller blocks, which would be a whole lot better.

I took a quick look for some sort of control in Windows XP that would increase the GDI resource memory pool, but didn't find one. I'll post something again if I find it later. But if you should find it, you might try increasing it as a short term workaround. I remember seeing this sort of thing in older versions of Windows.

Karl